Robust Error Correction in Infofuses

Robust error correction in infofuses

Greg Morrison

School of Engineering and Applied Sciences,

Harvard University, Cambridge, MA 02138

Sam W. Thomas III

Department of Chemistry and Chemical Biology,

Harvard University, Cambridge, MA 02138 and

Department of Chemistry, Tufts University, Medford MA 02155

Christopher N. LaFratta

Department of Chemistry, Tufts University, Medford MA 02155

Jian Guo

Department of Electrical Engineering,

Tufts University, Medford MA 02155

Manuel A. Palacios

Department of Chemistry, Tufts University, Medford MA 02155

Sameer Sonkusale

Department of Electrical Engineering,

Tufts University, Medford MA 02155

David R. Walt

Department of Chemistry, Tufts University, Medford MA 02155

George M. Whitesides

Department of Chemistry and Chemical Biology,

Harvard University, Cambridge, MA 02138

L. Mahadevan

School of Engineering and Applied Sciences,

Harvard University, Cambridge, MA 02138

Abstract

An infofuse is a combustible fuse in which information is encoded through the patterning of metallic salts. The constraints and advantages and unique error statistics of physical chemistry require us to rethink coding and decoding schemes for these systems. We take advantage of the non-binary nature of the our signal with a single bit representing one of $N = 7$ states to produce a code that, using a single or pair of intensity thresholds, allows the recovery of the intended signal with an arbitrarily high recovery probability given reasonable assumptions about the distribution of errors in the system. An analysis of our experiments with infofuses shows that the code presented is consistent with these schemes, and encouraging for the field of chemical communication and infochemistry given the vast permutations and combinations of allowable non-binary signals.

Infochemistry is an emerging field, attempting to develop methods of storage and transmission of information using chemical or material means. The advantage of these modes of communication over electronic communication will depend on the speed, reliability, and versatility of the transmission, as well as the conditions under which the signal is to be sent (e. g. with no power source available), but remains relatively unexplored, since the maximum rate at which information can be transmitted is limited by the physical length scales and timescales in the system as well as the noisiness of the channels which clearly differs significantly from an electronic analogue.

Recent work has shown that it is possible to transmit and receive messages using both chemical and material means, using an infofuse [1, 2], a combustible system in which patterned metallic salts encode information, and an infobubble [3], a bubble-based microfluidic device in which optical pulses encode information. Reliable transmission of a signal in either of these examples, and indeed in any communication system requires the development of methods to overcome the noise in transmission, which is itself a function of the system. Over the past half century, various error correcting codes [4–9] have been developed and allow for the possibility of accurate signal recovery with minimal loss of information density, for both binary and non-binary systems. In its most elemental form, error correction is accomplished by including redundant check bits, which convolve the positions and values of each bit in the signal in a simple manner. While binary communication schemes are the most commonly studied[5, 16], non-binary alphabets (where each bit can attain one of N possible values) may increase the efficiency of these redundant bits in correcting errors[9, 18]. However they might introduce further complexity in both encoding or correction. Here we focus our attention on the infofuse which uses a triplet code of pulses with a bit taking on $N = 7$ possible values and develop a simple error correcting code tailored for correcting the errors that takes advantage of the inherent non-binary nature of the system[9, 18]. The coding scheme introduces redundancy in the transmitted signal, with a message of length $n + m$ transmitted to communicate an intended signal of length n using m redundant check bits. Our approach which combines theory and experiment shows that this code can recover the intended sequence with an arbitrarily high probability given some simple but reasonable assumptions about the distribution of insertion errors. We further show that using a pair of thresholds, dividing the data into 'clear' and 'indeterminate' peaks can increase both the reliability of recovery and reduce the computational complexity. Finally, we show that for

the infofuse with a bit taking on $N = 7$ possible values, we achieve a good balance between the length of the fuse required to send a message and the efficiency (the ratio $n/(n+m)$) of the error correction.

THE INFOFUSE

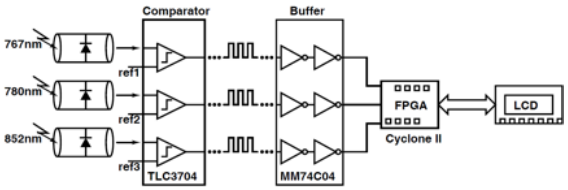

FIG. 1: Block diagram of portable detection system

The experimental system

Infofuses[1] are strips of a flammable polymer (nitrocellulose) patterned with spots of thermally emissive salts. An ignited infofuse supports a flame front that propagates along the strip at a roughly constant velocity and successively ignities each spot in turn, thereby emitting optical pulses in time. Infofuses use three distinct alkali metals with very sharp emission spectra: Potassium (K, at 767 nm), Rubidium (Rb, at 780 nm), and Cesium (Cs, at 852 nm). The nitrocellulose strips are on the order of 2-3 mm wide, and 0.1 mm thick. The speed of the flame front as it propagates along the fuse depends strongly on the fuse width, but is generally in the range of 1-3 cm/s. The emission from each of the chemical spots is observed by a telescopic receiver, which monitors the emission midpoint of each element. The observed light pulses have a duration of about 100 ms ($\sim$ 0.3 mm wide), and the spacings between subsequent chemical spots is on the order $\sim$ 1cm. The detector has excellent range, and clear signals are obtainable from more than 500m away (with an estimated 1.4km maximum range). Signals sent within the lab (close range of $\sim$ 20m only allowed 4-5 pulses per fuse before falling out of the range of view of the telescope, so long messages sent within the lab were broken into multiple fuses. For the purpose of encoding information with these three emitters, we used a scheme that assigns alphanumeric characters to simultaneous combinations of unique optical pulses[1]: seven $(2^3 - 1)$ unique

optical pulses exist for three distinct emitters. Each pulse combination is given a numerical value, with K=0, Cs=1, Rb=2, K/Cs=3, K/Rb=4, Cs/Rb=5, and K/Cs/Rb=6. Two consecutive pulses (giving a total of 49 unique pair combinations) are therefore sufficient to encode each alphanumeric character and some special characters (see the Supplementary Information for further discussion).

A portable infofuse detection system, in the form of a three-channel hard-limiter receiver, was implemented for long-distance experiments to verify the single threshold encoding/correction algorithm. The main modules of the system consist of three channels of high-sensitivity photo-receivers (PDF10A, Thorlabs Inc.) and peripheral optics that separate and amplify three spectrum emissions from burning infofuse (767nm, 78nm, and 852nm). The photo-receivers output signals are digitized by voltage comparators (TLC3704, Texas Instruments Inc.) using a single threshold , which can be independently adjusted (with V_i the voltage threshold for the i^{th} channel) to compensate for the differing sensitivities and incident intensities in each of the channels. The digital outputs of the comparators are then transmitted via digital buffers (MM74C04, Fairchild Semiconductor Corp.) to a high-speed Field-Programmable Gate Array (FPGA) chip for digital signal processing, where an asynchronous algorithm implements message acquisition and decoding. The algorithm puts the digital system into standby mode and only wakes it up for data acquisition if the optical-electrical signal of any of the three photo-receivers crosses the thresholds $V_1 - V_3$. In this way, the data acquisition rate can be automatically adjusted to fit the speed of burning in the infofuse, thus improving the data transmission efficiency and reducing system power consumption. The results of the decoded messages are displayed on a liquid crystal monitor (CFAH1604A-YYH-JT, Crystalfontz).

Errors in the infofuse

In Fig. 2(a) we show a typical sample of the measured intensity of the infofuse associated with the transmission of the message (in this case, the encoding for 'tufts', described further below) as a function of time, with sharp, well defined intended peaks whose intensity is variable, but well above the background noise over most of the signal duration. However at the beginning and end of this particular transmission the noise is much larger (typically the higher noise initially may be due to the ignition of a match); indeed decreasing the

thresholding level would likely add spurious additional Potassium peaks to the beginning and end of the signal and produce insertion errors in the signal. In Fig. 2(b), corresponding to the message we see that the intended peak near 8s (marked as intended in the figure) has a temporal profile similar to the other intended peaks, but with a significantly reduced maximum intensity. This is likely caused by either a misapplication of the metallic spot (too little Potassium), or because portions of the pattern did not successfully ignite. The intensities of the noise peaks near 8.5s and 9s are similar in both intensity and profile to the low intended peak at 8s, making it difficult to distinguish between signal and noise. While the peak near 8.5s may be discarded due to the small temporal spacing from its neighbors, reducing the spacing between peaks would make such a distinction more difficult. The noise peak near 9s could be discarded (as well below background), treated as an insertion error (a K/Rb peak followed by a temporally very close Cs peak), or merge with the intended peak (with the intended Cs peak read as an K/Rb/Cs peak). While the particular values of the thresholds in temporal spacing and intensity will have an effect on the noisiness of the signal sent by the infofuse, experimentally, we find that the primary sources of error are associated with the insertion of unintended bits and possible permutations of intended bits. Importantly, we do not have the problem of any *missing* peaks over the range of operation of our receivers, so that we do not need to worry about deletion errors as we develop a robust error correction scheme for the infofuse.

In order to overcome the noise in transmission, a simple solution is to simply increase the spacing between the metallic patterns. This will cause all of the noisy peaks to be well separated, so that unambiguously determining the applied chemicals is virtually assured without confusion due to neighboring peaks. However, this will decrease not only the physical rate of the information transfer, but also the length of the signals that can be sent (since messages must be encoded on a fuse of finite length). If we have n intended peaks separated by a distance d_0, and the average flamefront velocity is v, the physical rate is $r_0 = v/d$, with a time $\sim n/r_0$ required to transmit the message. By increasing the spacing from d_0 to d, we decrease the physical rate of order nd_0/d. In Fig. 2, the temporal separation between intended peaks is on the order of the width of the chemical patterning. Increasing the spacing by a factor of 2 will significantly separate the peaks, simplifying the process of distinguishing between data and noise, but at the cost of cutting the physical rate in half and double the length of the transmission time.

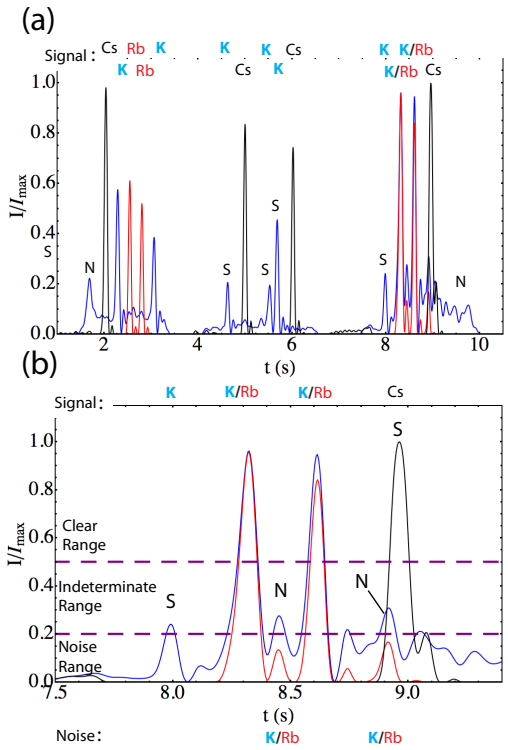

FIG. 2: Errors in the Infofuse. Blue represents K, black Cs, and red Rb. (a) shows a sample of the errors occurring for the infofuse. Noise peaks at the beginning and end of transmission have a somewhat broader profile, but peak intensities well above background. The intensity of the noise near 2s is on the order of the intended peaks near 4.5-5.5 s. Two peaks occur near 5.5s, with a temporal spacing smaller than expected. (b) Intended peaks may be difficult to distinguish between noise peaks in both maximal intensity and intensity profile. The noise peak near 9s could be considered either an insertion (noise followed by intended) or permutation (noise and intended giving a K/Rb/Cs signal).

An alternative to simply increasing the distance between peaks is to introduce a self-correcting code into our encoding for the infofuse, which will allow the correct signal to be reconstructed in the presence of noise. Such a code will be preferable to simply increasing the distance between peaks if it is more efficient (i.e. allows a message to be reliably sent at a higher rate). A variety of error correcting codes have been developed to allow the recovery of a noisy signal under differing noise conditions. In a perfect world, an error correcting code would be designed so that any code word sent could not be confused for any other possible code word, regardless of the errors that occur (see Fig. 3). The classic Hamming[4, 16]

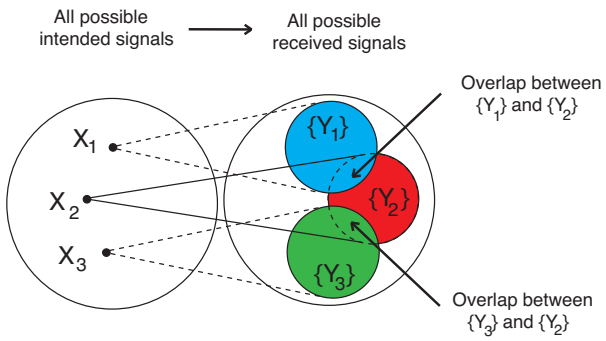

FIG. 3: Conceptual diagram of error correcting codes. Each intended signal X_i that can be sent will map onto a set of received signals $X_i \to \{Y_i\}$ that differ from the original. By restricting the sent messages to those whose mappings $\{Y_i\}$ do not overlap for all i, the intended signal can be perfectly reconstructed. If the overlap is small, the correct signal can only be recovered with high probability.

or Golay[6] codes, the more commonly used Reed Solomon codes [7], or the more modern and extremely high rate low density parity check codes[15, 19] allow the correction of up to a fixed number of permutation errors, $\lfloor (h-1)/2 \rfloor$ (with h the minimum or Hamming distance of the code), with 100% probability. All of these codes are effectively implemented for the correction of permutation errors, where the intended bit value d_i may be permuted via channel noise into an unintended value d'_i. However, experimentally we find that the infofuse suffers from insertion errors (where bits may be added to the signal, see Fig. 2), which can not be handled using these well known codes. While error correcting codes that address insertion or deletion errors have been studied [9, 11–13] from a variety of approaches, many are not optimally adaptable to non-binary codes. Low density parity check (LDPC) codes, which are extremely efficient for long signals, are somewhat inappropriate for the short signals that must be sent via the infofuse, due to the physical constraints on the length of the fuse. Additionally, codes which are capable of correcting insertions *and* deletions, while only insertions are observed experimentally, are expected to be less efficient than a code that focuses solely on insertion errors.

CODING SCHEME

Check Bits

We wish to develop a simple scheme that uses the non-binary nature of the infofuse to correct an arbitrary number of errors with high probability[19]. As seen in Fig. 2(a), if we choose a single, sufficiently low intensity threshold we can be certain of correctly including all data peaks, but must accept that some noise peaks will be inserted. Alternatively, if we choose a pair of thresholds with one sufficiently high, we can be sure that all noise peaks are excluded from the signal, but some data peaks may be excluded as well. These dropped data peaks will join a set of *indeterminate* peaks: intensity peaks that clearly indicate that there was some chemical present (well above background), but whose intensities are not high enough to be deemed 'clear' data peaks. In this case, the receiver must be able to accurately which of the indeterminate peaks were intended, and which are noise.

To allow error correction in the infofuse, the sender and receiver agree on the number of data bits, n, and the number of check bits, m, beforehand. The check bits are chosen to convolve both the position and the value of each data bit in a unique way. The check bits are chosen such that the first $N-1$ bits have the simple form

$$c_k = \sum_i i^{k-1} d_i \bmod N, \tag{1}$$

where d_i is the data bit value at the i^{th} position and $N = 7$ is the number of possible bit values. This simple form for the check bits is suboptimal in many respects, chosen only for clarity, simplicity, and flexibility (see the SI for further discussion). It is not difficult to see that the probability of a randomly chosen sequence producing a given set of $\{c_k\}$ is $p_{fail} = N^{-m}$ (similar to the discussion in Sec. 8 of Ref. [9]). This exponentially decaying probability will allow the determination of a robust error correcting code. Below, we describe the correction scheme for insertion errors (see the SI for discussion of permutation errors). While codes which have the ability to correct a single insertion or deletion error with 100% probability convolve data and position with two constraints[13, 14], we will see the use of multiple check bits and large-alphabet ($N = 7$) encoding allows us to correct an arbitrary number of insertion errors with high probability.

Single Threshold correction

If a single intensity threshold is used (see Fig. 2), with the cutoff set sufficiently low, the received signal will be a combination of all $n+m$ data and check peaks, as well as $k \geq 0$ noise peak insertions. For the moment, we assume the m check bits are perfectly recovered, and unrealistic and severe approximation that will be discussed in detail below). The possibility of permutation errors is also neglected here (see the SI for further discussion). In order to model the noisy transmission, we assume a uniform probability p that a noise peak is inserted following any observed peak in the data block. The number of insertion errors observed in the system then has the distribution $P_{ins}(k) = \binom{n+k+1}{k} p^k (1-p)^n$, giving $\langle k \rangle = np/(1-p)$, with a variance $\langle k^2 \rangle - \langle k \rangle^2 = np/(1-p)^2$. A receiver who finds a signal with $n+k$ peaks can simply determine all possible subsequences of length n and compare the received check bits to the computed values (where the final m bits are assumed correct, see below for further discussion). There will be $\binom{n+k}{n}$ such sequences, so long sequences with multiple errors may have an extremely high computational cost. The probability of recovering only the correct signal given k insertion errors is $P_{rec}(k) \gtrsim (1-N^{-m})^{\binom{n+k}{n}-1}$, and the total recovery probability (i.e. the probability that a single, unique sequence is recovered) is

$$P_{rec} \gtrsim \sum_{k=0}^{\infty} P_{ins}(k) \left(1 - N^{-m}\right)^{\binom{n+k}{n}-1} \tag{2}$$

The recovery probability as a function of m is shown in Fig. 4, and it is clear that P_{rec} is sigmoidal in nature and increases rapidly beyond the midpoint of the transition. This expression incorporates the total number of possible trial sequences rather than unique trial sequences (neglecting any correlations between the trial sequences), and thus will underestimate the probability of recovery.

A sequence containing the average number of insertion errors, $k = \langle k \rangle = np/(1-p)$, will require $\binom{n+k}{n} \sim \exp[nH_e(p)/(1-p)]$ trial sequences, where $H_e(p) = -p\ln(p) - (1-p)\ln(1-p)$, similar to the binary entropy function[4]. Due to the exponential growth of the computational complexity of the code, in practical implementations we must choose the number of data bits n such that the number of expected trial sequences is not too large. We expect the number of errors to scale as $k_0 \approx \langle k \rangle = np/(1-p)$, with higher order corrections scaling as $\delta k_0 \propto \sqrt{\langle k^2 \rangle - \langle k \rangle^2} \sim \sqrt{np}/(1-p)$, where the proportionality

constant is of order unity. It is not difficult to see that $P_{rec}(k_0 + \delta k_0) \sim 1 - \epsilon$ when $m = m_0 \sim -\log_N \left[1 - (1-\epsilon)^{\binom{n+k_0+\delta k_0}{n}^{-1}}\right]$, yielding

$$\begin{aligned} m_0 &\sim n\frac{H_N(p)}{1-p} - \sqrt{n}\frac{\log_N(p)\sqrt{p}}{1-p} + g(n,\epsilon) \\ H_N(p) &= -p\log_N(p) - (1-p)\log_N(1-p) \end{aligned} \tag{3}$$

with $g(n,\epsilon)$ an undetermined function satisfying $g(n,\epsilon)/\sqrt{n} \to 0$ as $n \to \infty$. g can in principle be determined numerically, but is analytically difficult to determine directly. This argument determines the number of check bits required to recover from $k_0 + \delta k_0$ errors with high probability, and essentially estimates the number of errors that may occur while the overlap in output sequences remains small (as diagrammed in Fig. 3). The asymptotic values of m are shown in Fig. 4(a) for various values of p, and the leading order term in n roughly coincides with the midpoint of the transition between low and high recovery probability. As $n \to \infty$, the dominant contribution to m will be the number of check bits required to reach the transition.

Higher rates using multiple thresholds

Improved recovery statistics can be attained by using a pair of thresholds, dividing the signal into clear, indeterminate, and background ranges. Peaks in the indeterminate range will be a mix of intended peaks and noise peaks, and the error correction scheme must be able to distinguish between the two. We assume the threshold I_{cut} is chosen sufficiently high so that no noise peaks ever fall into the clear range, else our coding scheme would fail. The receiver will find $n-l$ clear peaks, as well as $k+l$ indeterminate peaks (with l the number of intended peaks that are considered indeterminate and k the number of insertions). In order to recover the intended signal, all possible sequences of length n containing all $n-l$ clear peaks and l of the indeterminate peaks can be generated, and compared to the check bits. Again, this decoding scheme has high computational complexity for long, noisy signals, with $\binom{k+l}{l}$ trial sequences being generated. However, the number of trial sequences using a pair of thresholds is strictly less than when a single threshold is used.

We assume that the probability of an intended peak being considered indeterminate is uniform with probability q, and maintain the stringent assumption that all m check bits are

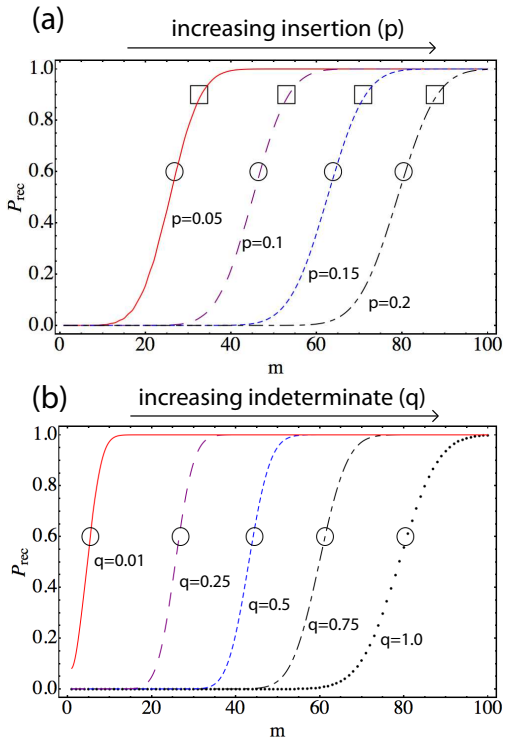

FIG. 4: Recovery probability for insertion errors. (a) shows P_{rec} as a function of the number of check bits m for varying insertion probability p, all with $n = 250$ (solid red, $p = 0.05$, dashed purple 0.1, dotted blue 0.15, and dash-dotted black 0.2). The large circles are centered on the asymptotic value of $m = nH_N(p)/(1-p)$, while the large squares are centered on the higher order solution in Eq. 3. (b) shows the recovery probability using a pair of thresholds with $p = 0.2$ and $n = 250$ for varying m and q (q being the probability of an intended peak found below the clear threshold). Shown are $q = 0.01$ (solid red line), 0.1 (dashed purple line), 0.25 (dotted blue line), 0.5 (dash-dotted black line), and 1 (black points). $q = 1$ is identical to the $p = 0.2$ curve in (a). Large circles denote the predicted midpoint value m_0 in Eq. 5.

perfectly recovered. The probability of recovering the the intended signal uniquely is then

$$P_{rec} = \sum_{k=0}^{\infty} \sum_{l=0}^{n} P_{ins}(k) P_{del}(l) \left(1 - N^{-m}\right)^{\binom{k+l}{l}-1} \tag{4}$$

with $P_{del}(l) = \binom{n}{l} q^l (1-q)^{n-l}$ and $P_{ins}(k)$ the uniform probability of insertion. Eq. 4 counts all possible trial sequences, not just unique trial sequences, and thus underestimates the probability of recovery. The average number of indeterminate peaks will be $\langle k + l \rangle = nq + np/(1-p)$, and we can perform the same analsis as in the single threshold case to

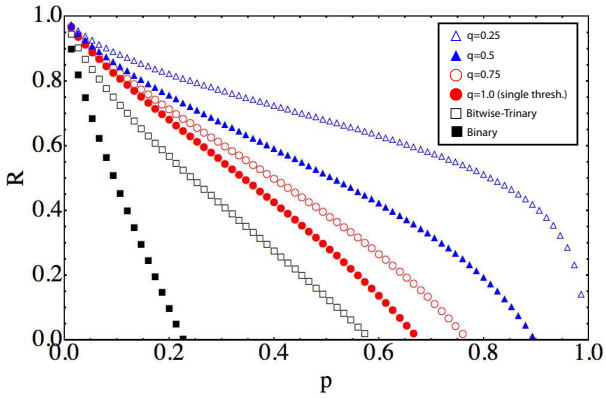

FIG. 5: The predicted information rate using meta-check bits from Eq. 6. Shown is the rate in the indeterminate channel with q=0.25 (blue empty triangles), 0.5 (blue filled triangles), 0.75 (red empty circles) and 1.0 (red filled circles) for $N = 7$. $q = 1.0$ is equivalent to the single threshold corrector. The filled black squares show the rate for $q = 1$, but for the binary $N = 2$, and the filled black squares show the rate for the trinary encoding with a division of bits into bytes of length $b = 2$ ($N_{byte} = 9$). These both clearly display the increased efficiency of large alphabet encoding.

estimate the number of bits required:

$$m_0 \sim \frac{n}{1-p}\left[(p+q-pq)\log_N\left(\frac{p}{1-p}+q\right) - p\log_N\left(\frac{p}{1-p}\right) - (q-pq)\log_N(q)\right] + O(n^{\frac{1}{2}}) \quad (5)$$

Eq. 5 is equivalent to Eq. 3 if the probability of dropping an intended peak $q = 1$, and decreases with decreasing q. In Fig. 4(b), we see that for fixed insertion probability p, the recovery probability has a sharper transition for far lower m as q decreases (see SI Fig. 1 as well). For decreasing q, the indeterminate error correction is becomes more reliable than the insertion correction. A pair of thresholds not only decreases the computational complexity, but also greatly increases the reliability of communication.

Meta-check bits

In our determination of the number of check bits required to correctly recover the intended sequence, we have thus far made the stringent assumption that the check bits are perfectly recovered. The results presented above can be used with minimal modification if we can be sure of recovering the check bits with high probability. This immediately suggests the use of meta-check bits, which perform the same redundancy on the check bits that the check

bits perform on the data. The meta bits will have the form $c'_k = \sum_i i^{k-1} c_i \mod N$, much like in Eq. 1. Each of these meta-check bits can suffer from insertion or indeterminate errors as well, so an additional set of bits must be used to check all meta bits as well. The multi-layered level of protection can be continued indefinitely (discussed further in the SI), allowing for an iterative protection scheme ensuring reliable recovery.

In the limit of large n, the k^{th} meta-check block will require $m_k/n = (m_0/n) \times (m_{k-1}/n)$ bits (with m_0 given in Eqs. 3 or 5).

$$m \sim n \sum_{k=1}^{\infty} \left(\frac{m_0}{n} \right)^k = n \, \frac{m_0}{n - m_0} \tag{6}$$

The rates $R = n/(n+m) = 1 - m_0/n$ of these codes are shown in Fig. 5. The rate vanishes at a finite value of p for all q, due to the fact that a sufficiently noisy transmission would require $m_0 > n$. At worst (with $q = 1$ in the single threshold case), the rate vanishes at $p_{max} \approx 0.68$. Interestingly, for a binary channel with $N = 2$, we find $p_{max} \approx 0.22$, showing the increased efficiency due to the non-binary coding. The indeterminate error correction scheme has an even higher rate as q (the probability of an intended peak being considered indeterminate) decreases.

It is worth noting that an alternative to our large alphabet encoding ($N = 7$) could be replaced with a binary or trinary system of individual peaks (i.e. K=0, Cs=1, Rb=2), which are grouped together in bytes of length b. Using 3 elements, each byte can attain $N_{byte} = 3^b$ states, so it is useful to determine the efficiency of our error correcting code using such an encoding system. A system which uses $b = 2$ (i.e. $N_{byte} = 9$) would still require a pair of bytes to represent an alpha-numeric character, but would require twice as many peaks as the $N = 7$ case to send the message. Eq. 3 can be adapted in a straightforward fashion, and we find that $p_{max} \approx 0.59 < 0.68$ for the byte-wise trinary signal. Such a method is thus less efficient than the large alphabet encoding used, despite the fact that $N_{byte} = 9 > N = 7$, both in the expected rate of the code (Fig. 5), as well as in the required fuse length to send a message ($b = 2$ transmission would require twice the length of fuse to send the same signal). Bytes grouped with $b \geq 3$ do allow a higher information rate than the large alphabet $N = 7$ encoding, but are more inefficient in terms of fuse length (with a $b = 3$ encoding having $p_{max} = 1$ but requiring 1.5 times the number of bits to send the message). The large alphabet encoding presented here thus strikes an excellent balance between length efficiency and error correction efficiency, important in the context of infochemistry.

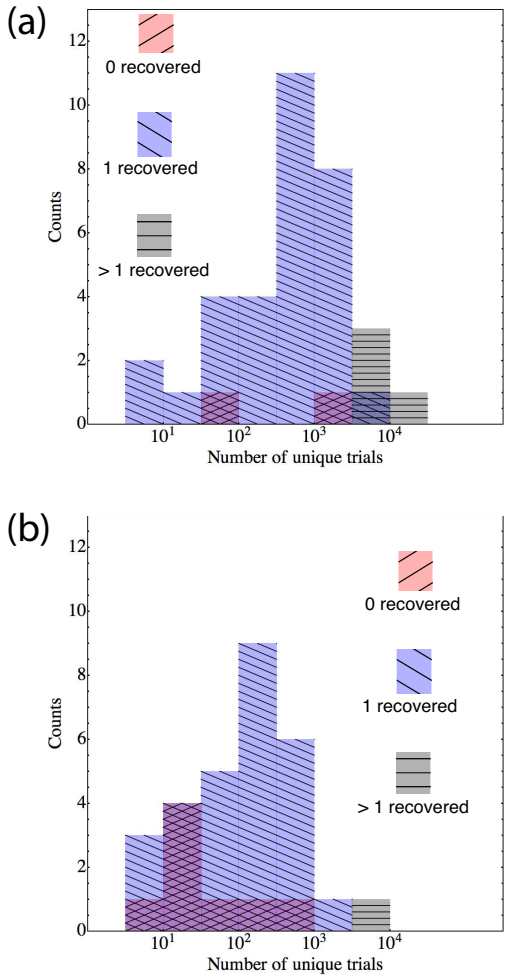

FIG. 6: The error correction of the message 'tufts,' using three check bits and one meta-check bit ($n = 10,\ m_0 = 3,\ m_1 = 1$). (a) shows the recovery using the $I_{cut} = 0.6I_{max}$, (b) shows recovery using $I_{cut} = I_{max}$. Both include the results correcting for 0, 1, and 3 permutation errors. Upward-slashes denote failed recovery, downward slashes denote unique recovery, and horizontal slashes denote multiple recovered sequences. Shown are only those sequences that gave fewer than 2×10^4 unique trials, which excludes 7 (14%) of the attempts from (b).

EXPERIMENTAL RESULTS

In order to test the applicability of our coding scheme to the experimental system, we sent the signal 'tufts' 19 times using 3 check bits and 1 meta check bit (14 total bits, with a rate $n/(n+m) = 0.71$, the encoding scheme is presented in the SI). The intended signal is 10220010010441, and translates to 'tuftsiz' with the four additional error-correcting bits. For each experiment, the intensity of each channel (K, Rb, and Cs) was rescaled

by the maximum intensity. Each message was decoded with a noise threshold of at most $I_{noise} = 0.12 I_{max}$, but the threshold was increased if necessary so that no more than 5 insertion errors occurred in any signal (an increase to $I_{noise} = 0.15$ was sufficient in all cases). For most signals this threshold was unnecessarily low, although such a low threshold was required for some. This threshold was applied to all experiments to be absolutely certainty that all data peaks would be included. A higher threshold would have produced fewer channel errors by ignoring more of the inserted peaks, so error correction with this threshold demonstrates a worst-case scenario. Peaks were considered to occur simultaneously if their maxima were within 0.08s, and peaks separated by more than 0.11s from both of their nearest neighbors would be included in the returned sequence (with all other peaks discarded as noise). The temporal thresholding is also lower than is strictly necessary, since the average spacing between intended peaks is ~ 0.3s. The use of a lower threshold again ensures that no intended peaks are dropped, at the cost of additional insertions or permutations. Because of the limited field of vision of the telescopic detector over short ranges, the signal was divided into three fuses, with 5, 5, and 4 bits respectively. This led to increased separation between blocks of bits (see Fig. 2(a)). The very limited range of temporal cutoffs (a range of 0.03s for discarding peaks as noise) suggests that this increased spacing does not significantly alter the noise statistics. When the two thresholds were used, the threshold for clear peaks was set to $I_{cut} = 0.5 I_{max}$ (see Fig. 6(b)). In all cases, we first tried to correct the signals without regard for permutation errors, but often found that the signal was not accurately recovered. We then corrected for a single permutation error (see the SI for discussion of permutation correction), which ensured the recovery of the correct sequence in all experiments.

In Fig. 6, we show histograms of the results of the error correction, using a single (Fig. 6(a)) or pair of thresholds (Fig. 6(b)). In both, we show the results when correcting for both zero and one permutation error in the same histogram. We divide the experimental results into attempted recoveries for which no trial sequences satisfied the check bits (red, upward slashes), those that recovered a unique sequence (blue, downward slashes), and those for which multiple sequences were recovered (grey, horizontal slashes). This histogram is plotted as a function of the number of unique trial sequences generated, rather than the $\binom{n+k}{n}$ sequences expected from the random coding arguments above. The average results are also listed in Table 1. We note that in all but one case, if any signals were recovered

Recovery statistics with two thresholds

perms	$\overline{\text{trials}}$	$\langle$correct$\rangle$	$\langle$recovered$\rangle$
0	33.9	0.58	0.58
1	202	1.0	1.16

Recovery statistics with one threshold

perms	$\overline{\text{trials}}$	$\langle$correct$\rangle$	$\langle$recovered$\rangle$
0	192	0.84	0.89
1	883	1.0	1.53

Table 1: Summary of the recovery statistics using one or two thresholds. The first column shows the number of permutation errors corrected, second column the geometric mean of the number of unique trials generated, third the fraction of trials that found the correct sequence, and fourth the average number of returned sequences.

one of them was the intended 'tufts'; however, other spurious matches were possible. In all cases where the unique sequence was not recovered, a permutation error had occurred but was uncorrected. In general, the number of unique trial sequences generated using a pair of thresholds was lower than the number using $q = 1$. This is due to the additional information created by labeling some peaks as 'certainly' intended, and greatly reduces the computational complexity of decoding. In Fig. 6, all occurrences of zero recovered sequences corresponded to at least one permutation error occurring. Correcting permutation errors can greatly increase the number of trial sequences that must be sampled, and increases the probability of finding more than one recovered sequences. The unique intended signal 'tufts' was recovered as long as the number of unique trial sequences was less than 3,418 (giving rise to three matches), while at worst 5 sequences were recovered (the intended and 4 spurious matches) for 21,015 unique trials. The largest number of trial sequences that resulted in a unique (and correct) match was 4,207. This shows the importance of correlations in the trial sequences, since for a randomly drawn set of trials, we would expect the probability of recovering a unique match as $P_{rec} < (1-7^{-3})^{4207} \approx 5\times10^{-6}$ (see Eq. 2). The random coding argument presented above significantly underestimates the probability of recovery, and the surprisingly small number of recovered sequences given the number of trial sequences shows the importance of correlations between trial sequences. It is worthwhile to note that even

in cases where multiple sequences satisfy all of the check bits, the error correction scheme still produces a drastic reduction in the number of possibly intended messages. In the worst case, with 21k trial sequences, the five recovered messages translate to 'tufts,' 'saets,' 'saosr,' 'tess3,' and '_efs*.' While it is clear that the possibility of multiple spurious matches is a limitation of our coding scheme, it is equally clear that the 21k possible sequences are reduced to a set of five, for which the english message is clearly distinguishable.

CONCLUSIONS

We have presented a relatively simple method for error correction for messages sent via a burning fuse patterned with metallic salts. Our coding scheme depends on the experimentally observed noise properties of the system: while noise peaks may cause insertion or indeterminate errors (by being indistinguishable from data peaks), no information is truly lost from the system. Even with the simplest possible representation for the redundant error-correcting bits (see the SI), the experiments show that our correction scheme is able to recover the intended signal in the presence of noise with high probability. By introducing layering in the error correction, in the form of the meta-check bits, it is possible to achieve arbitrarily high probability of signal recovery, assuming errors are sufficiently rare. A random-coding argument that shows there is an achievable, finite rate for error correction in an idealized infofuse system. However, the experiments show a much higher probability of recovery than would be expected given the random coding argument, due to the correlations in trial sequences.

The highly efficient nature of the code is due primarily to the non-binary nature of the encoding in the infofuse. A binary or trinary signal can be encoded by grouping bits into bytes of finite length in order to produce a large number of states per byte, but has an associated cost in increasing the required length of the signal. We have seen that not only does byte-wise trinary communication require effectively doubling the length of the signal, but also that the increase in the average number of errors reduces the efficiency of the code in comparison to a large N bit-wise coding. It is worthwhile to note that this code could be applied to a system where a larger N is used for encoding, such as increasing the number of emissive salts used in the preparation of the infofuse. Likewise, higher information density could be achieved by using the concentration of salts in each spot to encode information[1].

While the choice of appropriate threhsholds could be more difficult in this situation, we expect that increasing the alphabet size will give an increase in the efficiency in the code [18]

The particular nature of the noise observed in the infofuse, coupled with the great advantages in large N encoding, suggest that there may be great advantages in certain cases in chemical communication. One can envision a multitude of physical or chemical systems in which the number of 'bits' can represent a many states (large N) in a novel form of communication. For each, all that remains is a more complete understanding of the errors caused in transmission, and how to design an error correcting code that takes advantage of the noise characteristics for each. In some cases, the many well developed codes designed for binary communication may be adaptable or immediately applicable, but new techniques may be required to fully utilize the advantages of each system while simultaneously overcoming the inherent disadvantages in the same.

[1] Thomas SW, et al. (2009) *Proc. Natl. Acad. Sci* 106:9147.

[2] Kim, C, Thomas, SW, Whitesides, GM. (2010) *Angewandte Chemie* 122:4675

[3] Hashimoto, M, et al. (2009) *J. Am. Chem. Soc.* 131:12420

[4] Cover T, Thomas J (1991) *Elements of Information Theory* (Wiley, New York).

[5] Peterson WW, Weldon EJ (1841) *Error-Correcting Codes* (MIT Press, Cambridge, MA), 2^{nd} edition.

[6] Cohen G, Honkala I, Litsyn S, Lobstein A (1997) *Covering Codes* (Elsevier Science BV, Amsterdam).

[7] Wicker SB, Bhargava VK (1994) *Reed-Solomon Codes and their applications* (IEEE Press, Piscataway, NJ).

[8] MacKay DJC (2003) *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, Cambridge).

[9] Mitzenmacher M (2009) *Prob. Surveys* 6:1–33.

[10] Shannon CE (1948) *Bell System Tech. J.* 27:379.

[11] Davey MC, MacKay DJC (2001) *IEEE Trans. Info. Th.* 47:687.

[12] Klove T (1995) *IEEE Trans. Info. Th.* 41:279.

[13] Tenengolts G (1984) *IEEE Trans. Info. Th.* 30.

[14] Sloane NJA (2002) *Codes and Designs*, eds Arasu KT, Seress A (Ohio State Univ. Math. Res. Inst. Publ.), p 273.

[15] Gallager RG (1962) *IRE Trans. Info. Th.* 8:21.

[16] Hamming RW (1950) *Bell System Tech. J.* 29:147.

[17] Drinea E, Mitzenmacher M (2007) *IEEE Trans. Info. Theory* 53:2693.

[18] Mitzenmacher M (2006) *IEEE Trans. Info. Theory* 52:5496.

[19] MacKay DJC (1999) *IEEE Trans. Info. Theory* 45:399.

[20] Conway JH, Sloane NJA (1986) *IEEE Trans. Info. Theory* 32:41–50.